**Objective:**

Developing a camera based system that utilizes computer vision techniques to accurately track human legs in real time.

**Aim:**

To create an app which can do step counting without having contact.

**Workplan:**

* Get training videos from a few different angles
* Find a good model with CV techniques to accurately track human legs.
* Using the landmarks from the model, try out different ways to count the steps. (try with plotting the distance between legs with time)

**Try:**

I found some videos of people walking on treadmill online.

I tried the following models to track landmarks:

1. Cvzone: uses mediapipe and cv2 .
   1. Benefits: small code, easy to understand and start, works very well on clear videos.
   2. Drawbacks: fails miserably in front view and anything except side view.
2. Mediapipe:
   1. Benefits: can be fine tuned a little better than cvzone, can change the code to track only legs, works very well on clear videos
   2. Drawbacks: fails miserable in front view and anything except side view, once it landmarked my chair instead of my legs.
3. I am thinking of creating my own model using tensorflow which is trained on some videos and can detect legs. Then I can extract 30 frames per second from any video clip and send each image through this model and mark the landmarks.  
   Issues with it: This will take a long time and may give less accuracy.   
   I should try reading the source code of mediapipe before trying this.
4. There are several models present today to perform pose estimation. Some of the methods for pose estimation are given below:
   1. Open pose - multi person detection - we dont need it.
   2. Pose net or HR net {with PyTorch} - High accuracy, with some models like HRNet offering state-of-the-art results.
   3. Blaze pose {with mediapipe} - This is an introduction to「BlazePose」, a machine learning model that can be used with [ailia SDK](https://ailia.jp/en/). You can easily use this model to create AI applications using [ailia SDK](https://ailia.jp/en/) as well as many other ready-to-use [ailia MODELS](https://github.com/axinc-ai/ailia-models). It can be used for example in fitness applications.
   4. Deep Pose - DeepPose was proposed by researchers at Google for Pose Estimation in the 2014 Computer Vision and Pattern Recognition conference
   5. Dense pose - maps all the pixels on the body - redundant
   6. Deep cut - multi person pose estimation
   7. PyMO
   8. MoveNet {with tensorflow} - [MoveNet Model](https://github.com/tensorflow/tfjs-models/tree/master/pose-detection/src/movenet) is the latest pre-trained machine learning library [released](https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-movenet-and-tensorflowjs.html) by [TensorFlow](https://www.tensorflow.org/) team, as part of a larger [Pose Detection](https://github.com/tensorflow/tfjs-models/tree/master/pose-detection) [TensorFlow.js models](https://www.tensorflow.org/js/models) set.
5. The current opensource implementations include TensorFlow’s Tflite model, Openpose, and Pytorch models. Try implementing these

I will get a cvzone code working and running. It works well when the entire body of the person is clearly visible with good lighting. Once the algorithm is working fairly well with the step counting, then I will try to make a model with better tracking accuracy.

**TensorFlow’s TFLite MoveNet** is the best choice for single-person step counting using motion capture. It offers a perfect balance of real-time performance, high accuracy, ease of implementation, and deployment flexibility. While OpenPose and PyTorch models have their strengths, they either offer more complexity than needed or require more computational resources, making MoveNet the optimal solution for this specific application.

Read up on PyMo it looks good.

Algorithm for step counting:

| Title (with exp filter) | Dist btwn ankles | Dist between knees | Angle |
| --- | --- | --- | --- |
| Video 5 |  |  |  |
| Video 6 |  |  |  |
| Video 10 |  |  |  |
| Video 11 |  |  |  |
| Video 13 |  |  |  |
| Video 14 |  |  |  |
| Video 16 |  |  |  |
| Video 18 |  |  |  |
| Video 19 (Movenet) |  |  |  |

* Since the distance between the ankles data is the most uniform, I will proceed with counting the number of peaks in that.
* Basic idea 1: calculate the mean of first 5 points measured, multiply it by 5 and draw a horizontal line, the number of times that the graph crosses this line in positive direction is the number of peaks. <didnt work because the first 5 points arent always the start, sometimes it is the middle of the step and then the threshold becomes too high. May have to find a way to normalise it if that works.>
* Basic idea 2: from the ankles graphs we can see that there is atleast one peak befor 60 frames, so we first scan the first 60 frames (2 seconds for calibration), find the max of them all,find the min of them all and set the threshold to the midpoint between the two.
  + With exponential filter it will hopefully not count noise.
  + <temporarily works for video 5 test case, but need to find a better way to do this.>
  + Need to find a way to make sure that noise doesnt affect the peaks count and that the threshold isn’t fixed, it can change with time.
  + Found a way to change the threshold, everytime the values hit the threshold, I start finding the max and min of the peak and their average is the new threshold. To avoid a sudden change due to noise, I have put exponential filter on the updating threshold values.

Could use this dataset to train my own model:

The Carnegie-Mellon University dataset is a collection of about 2500 BVH files containing several types of movements. Simple movements such as jumping, walking, running to complex movements such as swordplay and swimming are present. The data, as well as the BVH hierarchy specified by the dataset, has set a standard for most research based on this field. The complexity and diversity of the data provide for adequacy in any training required.

* Leeds Sports Dataset (LSP): This dataset contains 11000 training and 1000 testing images from sports activities with challenging in terms of appearance and especially articulations. In this dataset, For each person the full body is labeled with total 14 joints.

Keeping the same accuracy of whatever we currently have, moving on to the next step, app dev.